

Fonctionnalité : Recherche	Fonctionnalité #3
Problématique : Rechercher la méthode la plus performance pour implémenter une fonction de recherche “temps réel” suite à une saisie de l'utilisateur. celle-ci doit démarrer dès le 3eme caractère tapé et afficher les recettes contenant les caractères saisis. La recherche s'effectue sur les champs titre, description et ingrédients	

Option 1 : Utilisation des fonctions natives de manipulation de tableaux Javascript (map, select, reduce, ...) Dans cette option, nous implémentons l'algo avec les fonctions natives censées être plus rapides. <ul style="list-style-type: none"> - Enumération des recettes avec la fonction map - Enumération des ingrédients avec la fonction 'reduce' 	
Avantages <ul style="list-style-type: none"> ⊕ L'algo est très compact ⊕ Utilisation des fonctions optimisées de JS 	Inconvénients <ul style="list-style-type: none"> ⊖ Code moins lisible ⊖ Perf dépendante de l'implémentation du navigateur
Mesures de performance: <ul style="list-style-type: none"> - Utilisation de JSBen.ch : voir la page https://jsben.ch/6j1NO - Test “in-situ” des deux fonctionnalités sur la branche git “mesure-des-deux-fonctionalites”, résultats dans la console 	

Option 2 : Utilisation des boucles “legacy” Dans cette option nous n'utilisons que les fonctions de base du Javascript pour faire la recherche <ul style="list-style-type: none"> - Boucle for pour énumérer la liste des recettes - Boucle for pour énumérer les ingrédients 	
Avantages <ul style="list-style-type: none"> ⊕ Code lisible ⊕ Perf peu dépendante de l'implémentation 	Inconvénients <ul style="list-style-type: none"> ⊖ Potentiellement moins performant
Mesures de performance: <ul style="list-style-type: none"> - Utilisation de JSBen.ch : voir la page https://jsben.ch/6j1NO - Test “in-situ” des deux fonctionnalités sur la branche git “mesure-des-deux-fonctionalites”, résultats dans la console 	

Solution retenue : Préférence pour le code le plus lisible (legacy) étant donné que JSBen.ch et le test in-situ montrent que les performances des deux solutions sont proches pour des textes de longueur courte. En tout cas dans le pire cas, la moins performante des deux solutions prend entre 70 et 200 microsecondes pour faire la recherche, ce qui est négligeable

Annexes

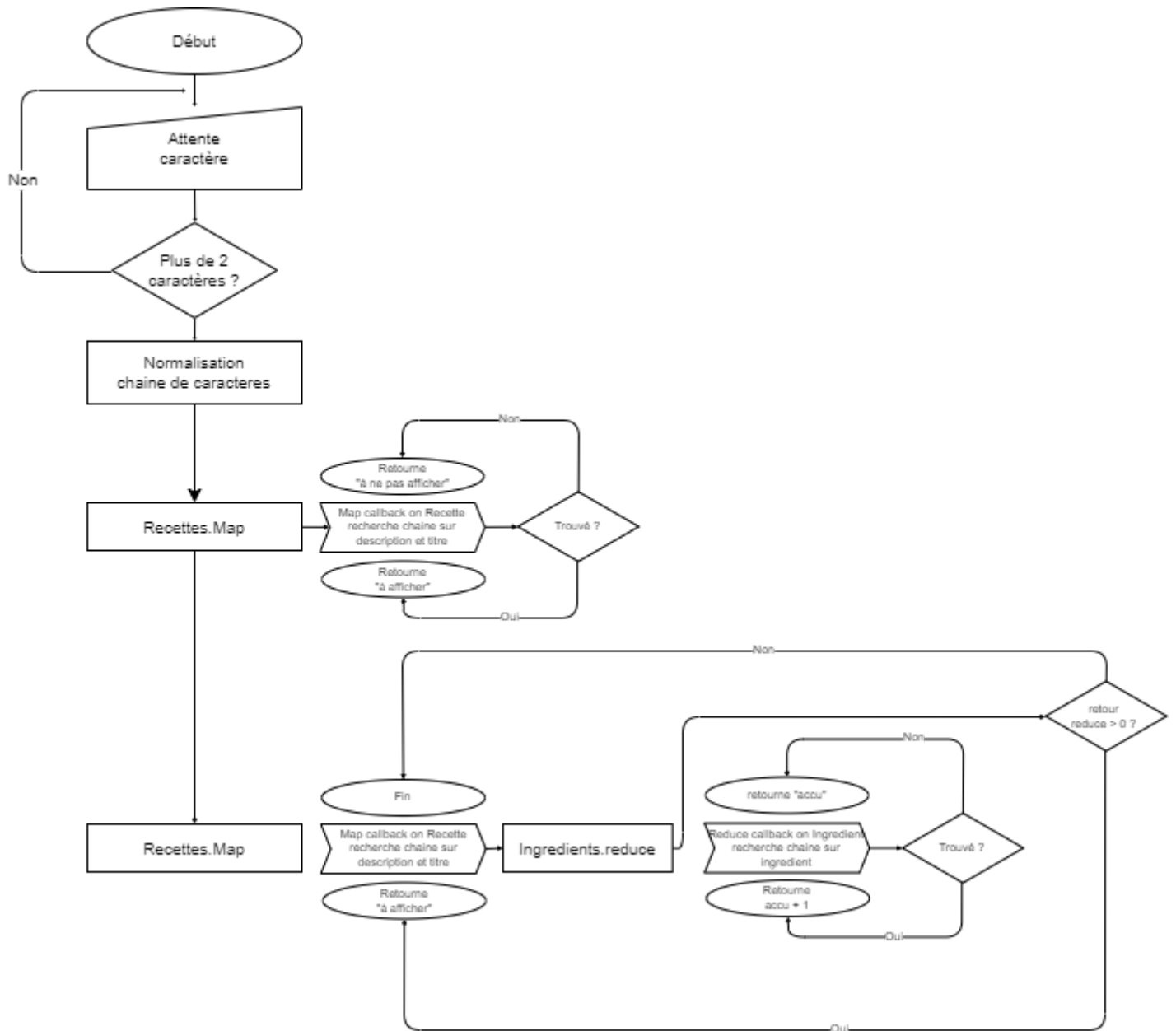


Figure 1 - Implémentation "legacy"

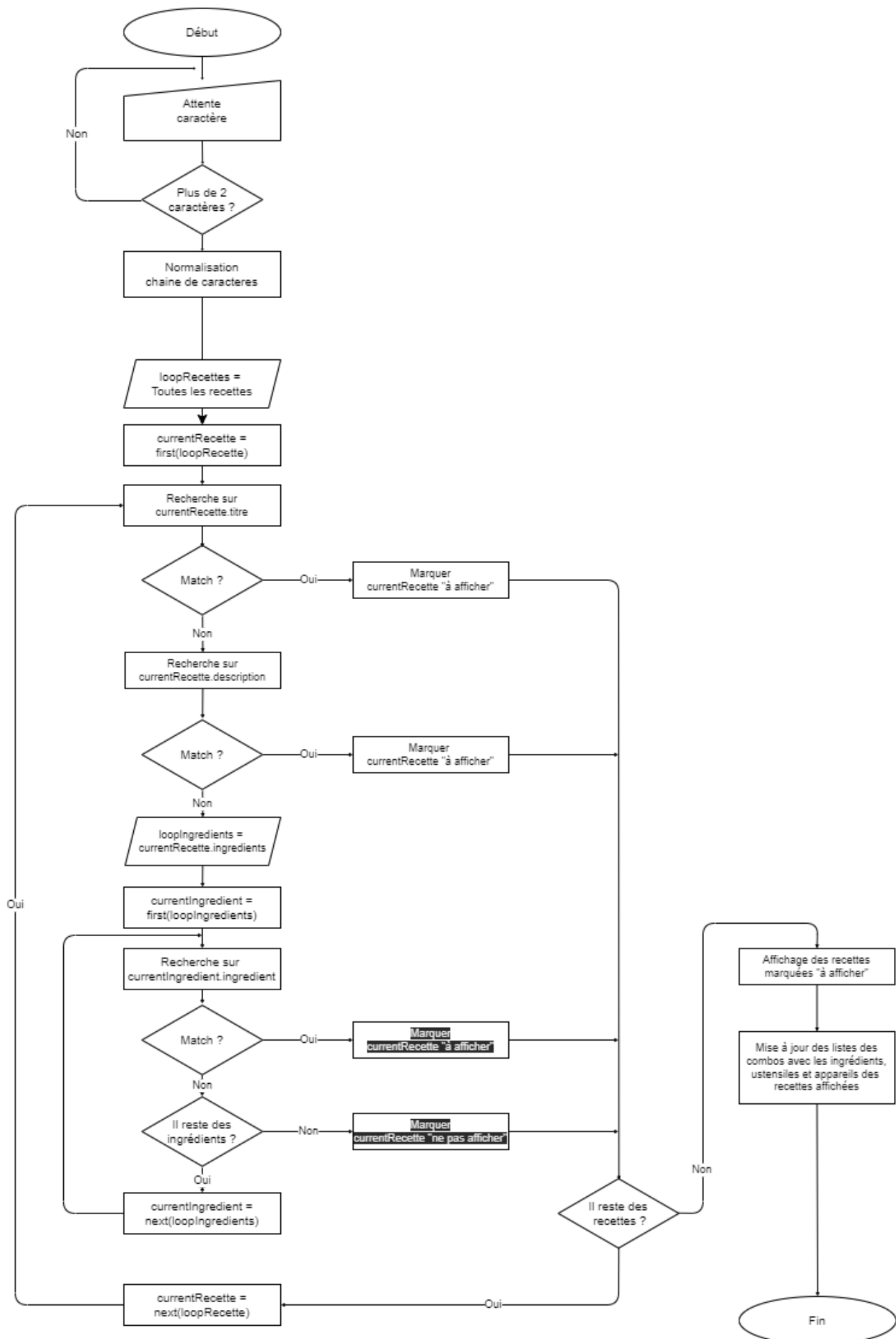


Figure 2 : Implémentation “array method”